

SMCL - Stochastic Model Checker for Learning in Games

Hongyang Qu, Michalis Smyrnakis and Sandor M. Veres

Department of Automatic Control and Systems Engineering
University of Sheffield, Sheffield S1 3JD, UK
{h.qu, m.smyrnakis, s.veres}@sheffield.ac.uk

November 23, 2016

Abstract

A stochastic model checker is presented for analysing the performance of game-theoretic learning algorithms. The method enables the comparison of short-term behaviour of learning algorithms intended for practical use. The procedure of comparison is automated and it can be tuned for accuracy and speed. Users can choose from among various learning algorithms to select a suitable one for a given practical problem. The powerful performance of the method is enabled by a novel behaviour-similarity-relation, which compacts large state spaces into small ones. The stochastic model checking tool is tested on a set of examples classified into four categories to demonstrate the effectiveness of selecting suitable algorithms for distributed decision making.

1 Introduction

Distributed decision making has recently received considerable attention for the development of autonomous agent teams, and in general, where agents need to make their decision in interaction with their environment, including other agents. Where autonomy is a desired property for a robotic agent application, for instance in remote or dangerous fields, battle fields, disaster sites, mining and nuclear waste processing applications, in such scenarios game-theoretic learning algorithms can be used as coordination mechanism between agents. Other applications are field robotics [18], wireless sensor networks [9, 10], smart grids [1, 23], disaster management [7, 20] and scheduling tasks [21]. Game theoretic learning can also be useful in complex environment such as autonomous urban transport, warehouses, airports and construction sites, as direct communication between agents could be blocked or broken and consequently the system has some random, stochastic behaviour.

When autonomy is a desirable property of an agent team then distributed decision making becomes an important attribute. Coordination mechanisms are needed to enable the agents to accomplish a task or achieve a goal. Many learning algorithms have been proposed in the literature for coordination mechanisms, which differ in communication requirements, computational cost and presence or absence of convergence proofs. Overall, there has however been no technique available to study collective behaviour when only a small number of iterations are allowed to achieve results. From an application point of view it is important to learn and accomplish some desired task quickly due to various constraints. This paper provides methods to study learning performance during short periods.

Apart from short time periods for learning, a second hindrance of effective cooperation can be the communication bandwidth and the amount of communication needed for collective achievement by an agent team. A game-theoretic formulation, where agents mostly observe each other, and communicate little, is favourable for many applications. The game theoretic cooperation we consider in this paper needs very little or no communication at all between agents. Lack of communication is compensated by the agent's observing each other and learning each others strategies.

A cooperative game-theoretic learning algorithm can be used to solve a single decision problem. In a complex task, where a sequence of decisions are needed, a sequence of algorithms can be applied. If this approach is applicable, then games in strategic form are generated for each type of decision making problem in a scenario of players, physical environment and rewards. This strategic form game can be seen as a snapshot of a general learning framework, which models the whole process of accomplishing the task. For example, partially observable games (POGs) are the game theoretic equivalent of decentralised Partially Observable Markov Decision Processes (dec-POMDPs) [3, 13, 15]. A technique for solving a POG is to generate strategic form games each time there is a change in the scenario of the world, i.e. a strategic form game is based on the scenario of the world, meaning the players, the physical environment and the shared goal and reward functions. In the snapshot, the scenario does not change any more, and players then choose actions to maximise their rewards [3, 4] using game-theoretic learning algorithms.

Traditional approaches to analysing the performance of a game-theoretic learning algorithm are usually based on convergence proofs (Nash and Pareto Nash optima) or based on extensive simulation studies. In simulation it is not always feasible or easy to understand the agents' behaviour. The amount of simulations needed can be huge, which can leave the possibilities of bad behaviour unexplored. This paper introduces and analyses a stochastic model checking method which bundles sets of state evolutions together to reduce complexity and hence makes probabilistic verification feasible and practicable to replace simulation studies.

The techniques presented explore the whole state space of a given learning problem. A state implies a probability distribution of joint actions by game players in a Discrete-Time Markov Chain (DTMC); a transition between two states represents a joint action associated with a firing probability. Such a DTMC-based state-space model can contain a large number of states due to the parameters used in some algorithms. We introduce a *behaviour similarity relation* “ \preceq ” among states, which enables efficient computation of steady state probabilities using probabilistic model checking [16] for *Nash Equilibria*. The performance of a learning algorithm can be measured by the probability at which a Nash equilibrium will be observed after a finite number of iterations. Nash equilibria are good measures of performance as they can be seen as optimal solutions to distributed decision making problems. However, our verification tool is not restricted to the cases where steady states exist. It is possible that a learning algorithm will not reach a steady state after a small number of iterations, or that a specific sequence of joint actions will lead to a steady state after a few iterations, while other sequences will not. The tool can be used to compute all these probabilities and analyse all the possible short term outcomes of the learning-algorithms within a given time period.

The structure of the paper is as follows. First our game theoretic framework is defined with learning, which is followed in Section 2.2 by simulation examples using Fictitious Play [2] on the coordination game of Equation (4) and the Shapley’s game [17]. The approach for computing finite probabilities for a game with initial conditions is presented in Section 3. We show comparison among all supported algorithms on several examples in Section 4, and conclude the paper in Section 5.

2 Preliminaries

2.1 Game-theoretic definitions

Definition 1 *A game in strategic form has the following elements [6]:*

- *A set of players, $i = 1, 2, \dots, \mathcal{I}$;*
- *A set A^i of actions for each player i , from which it selects its action a^i ;*
- *A set $A = \times_{i=1}^{\mathcal{I}} A^i$ of joint actions, where a joint action $a \in A$ is of the form $a = (a^1, a^2, \dots, a^{\mathcal{I}})$;*
- *A reward function $r^i : A \rightarrow \mathbb{R}$, where $r^i(a)$ is the reward that player i will gain if the joint action a is played.*

Given a joint action $a = (a^1, a^2, \dots, a^{\mathcal{I}})$, we often write $a^i \in a$ to indicate that a^i ($1 \leq i \leq \mathcal{I}$) is a component of a . We also refer to the set of joint

actions of all players but i as A^{-i} , and $r^i(a^i, a^{-i})$ the reward that player i will gain if he plays action a^i and its opponents play $a^{-i} \in A^{-i}$. Often we write $r^i(a^i)$, instead of $r^i(a^i, a^{-i})$, when a^{-i} is of no interest. Let Δ^i be the set of all the probability distributions over player i 's action space. A strategy $\sigma^i \in \Delta^i$ denotes the probability distribution player i uses to choose from its available actions, and $\sigma^i(a^i)$ the probability of choosing action $a^i \in A^i$. A *pure* strategy is the case where player i choose a single action with probability 1. Other cases are *mixed* strategies. A joint strategy is defined as $\sigma = \prod_{i=1}^I \sigma^i \in \times_{i=1}^I \Delta^i$. We often write σ^{-i} for a joint strategy of player i 's opponents. We write $\sigma(a^i, \sigma^{-i})$ for the probability of player i playing action a^i when its opponents use strategies σ^{-i} . By some abuse of notation, we use $r^i(a^i, \sigma^{-i})$ to denote the expected reward of player i when it chooses action a^i and its opponents use the joint strategy σ^{-i} , i.e.,

$$r^i(a^i, \sigma^{-i}) = \sum_{a^{-i} \in A^{-i}} r^i(a^i, a^{-i}) \sigma^{-i}(a^{-i}),$$

where $\sigma^{-i}(a^{-i})$ is the probability of playing the joint action a^{-i} by player i 's opponents. Similarly, for a strategy σ^i , we can write

$$r^i(\sigma^i, \sigma^{-i}) = \sum_{a^i \in A^i} \sum_{a^{-i} \in A^{-i}} \sigma^i(a^i) r^i(a^i, a^{-i}) \sigma^{-i}(a^{-i}).$$

To choose actions, players can use either deterministic or stochastic rules based on joint strategies. A common deterministic rule is *best response* (BR) $\hat{\sigma}$ in Equation (1) where players choose an action maximising their expected reward. Therefore the players are rational, as they always increase or maintain their reward, given that the other players are rational players too and the estimates of the other players' strategies are correct.

$$\hat{\sigma}^i = \operatorname{argmax}_{\sigma^i \in \Delta^i} r^i(\sigma^i, \sigma^{-i}) \quad (1)$$

Remark. Let us introduce some arbitrary mapping of A^i into $M := \{1, \dots, |A^i|\}$ where $|A^i|$ is the cardinality of A^i . Denote a generic element of M by j . In other words, the j is an indexing of the elements of M according to some arbitrary but fixed ordering. If a game has more than one best response, then we consider that players will always choose the action with the smallest index j .

Smooth best response $\bar{\sigma}$ in Equation (2) is the most common stochastic rule.

$$\bar{\sigma}^i(a^i, \sigma^{-i}) = \frac{\exp(r^i(a^i, \sigma^{-i})/\tau)}{\sum_{\tilde{a}^i \in A^i} \exp(r^i(\tilde{a}^i, \sigma^{-i})/\tau)} \quad (2)$$

where τ is a randomisation parameter.

With a relatively large τ , e.g., $\tau = 0.01$, smooth best response allows players to choose actions that do not maximise their expected reward with non-zero probability. When players use a small τ , e.g., 0.0001, they intend to choose with probability 1 the action that maximises their expected reward, which is equivalent to best response. A small τ is preferred in many practical applications because of its deterministic nature, which makes the game playing predictable. On the other hand, it reduces the chances of exploring different actions, which could lead to find better solutions either in few iterations or with high probability [12]. In the rest of the paper, we combine two decision rules together by using smooth best response with $\tau = 0.01$ in the first iteration and best response in other iterations. This is a very simple annealing scheme [11] to make a balance between probabilistic and deterministic behaviour.

Nash in [14] proved that every game has at least one equilibrium which is a fixed point in the best response correspondence. Thus, when players choose their actions using Equation (1), a strategy $\hat{\sigma}$ is a Nash equilibrium if

$$r^i(\hat{\sigma}^i, \hat{\sigma}^{-i}) \geq r^i(\sigma^i, \hat{\sigma}^{-i}) \quad \text{for all } \sigma^i \in \Delta^i, i = 1, \dots, \mathcal{I}. \quad (3)$$

Equation (3) implies that if the joint strategy $\hat{\sigma}$ is played in a game and it is a Nash equilibrium, then no player can increase its rewards by unilaterally changing its own strategy. When all the players in a game select their actions using pure (mixed, resp.) strategies, the equilibria are called *pure (mixed, resp.) strategy Nash equilibria*. An equilibrium is called *Pareto efficient* if no other joint mixed strategy can increase the payoff of all the players.

Example. Equation (4) shows a simple coordination game

$$r = \begin{matrix} & \begin{matrix} a_1 & a_2 \end{matrix} \\ \begin{matrix} b_1 \\ b_2 \end{matrix} & \left[\begin{array}{c|c} 1, 1 & 0, 0 \\ 0, 0 & 1, 1 \end{array} \right] \end{matrix} \quad (4)$$

In this example, player 1 can choose action b_1 or b_2 , and player 2 can choose a_1 or a_2 . In each row and column intersection the rewards of players 1 and 2 are listed. The maximum reward, and pure Nash equilibrium of the game, is 1 for both players when they execute the joint action (b_1, a_1) or (b_2, a_2) .

2.2 Basic game-theoretic learning algorithms

Iterative learning algorithms have been introduced as negotiation mechanisms between the agents. Typically, such an algorithm works as follows [5]. Each player repeatedly plays the game by trying to estimate other players' next move based on the observation of their behaviour history, and then selecting its own action to maximise its reward. They stop the iterative assessment if either they converge to an equilibrium or the maximum number of iterations is reached.

Fictitious Play [2] is the canonical example of game-theoretic learning. Each player i in the initial iteration has a non-negative weight function $\kappa_0^{i \rightarrow j} \in (0, 1)$ for the actions of each of its opponents j ($j \in \{1, \dots, \mathcal{I}\} \setminus \{i\}$) and update this weight function after observing player j 's action. Note here that the initial weights can be any arbitrary positive real number but this can significantly influence the estimations of opponents strategies and for that reason we choose to normalise the initial weights in order to be between zero and one. Therefore, we have

$$\sum_{a^j \in A^j} \kappa_0^{i \rightarrow j}(a^j) = 1. \quad (5)$$

Based on these weights, each player estimates its opponents' strategies and choose an action that maximises its expected reward. Let $\kappa_t^{i \rightarrow j}(a^j)$ be player i 's weight for player j 's action $a^j \in A^j$ at the t -th iteration, and $a_t^j \in A^j$ the action chosen by player j at the t -th iteration. At the t -th iteration, the weight is updated as follows.

$$\kappa_t^{i \rightarrow j}(a^j) = \kappa_{t-1}^{i \rightarrow j}(a^j) + \mathbb{I}_{a_{t-1}^j = a^j}, \quad (6)$$

where $\mathbb{I}_{a_t^j = a^j} = \begin{cases} 1 & \text{if } a_{t-1}^j = a^j \\ 0 & \text{otherwise.} \end{cases}$

Equation (6) simply increases the weight by 1 for the action executed in the previous iteration. Fictitious play is based on the implicit assumption that players use the same strategy in all iterations of the game. Based on this assumption, player i uses a multinomial distribution to estimate player j 's strategy $\sigma_t^{i \rightarrow j}(a^j)$ by the following formula:

$$\sigma_t^{i \rightarrow j}(a^j) = \frac{\kappa_t^{i \rightarrow j}(a^j)}{\sum_{a' \in A^j} \kappa_t^{i \rightarrow j}(a')}. \quad (7)$$

Player i chooses the action which maximises its expected reward, based on Equation (7) and (2). For Equation (2), $\sigma^{-i} = \times_{j=1}^{j=\mathcal{I} \wedge j \neq i} \sigma^{i \rightarrow j}$.

Remark. We should mention here that BR is a deterministic decision making rule in the sense that players always choose the action which maximises their expected reward. Nonetheless, it can lead to players using mixed strategies when a number of actions can be played interchangeably in the iterations of the game. If this is the case, the probability of choosing each of such actions is always greater than 0.

FP can be seen as the following three step process: (1) observe the opponents' actions; (2) update the beliefs about the opponents' strategies; (3) choose an action based on (smooth) best response decision rule. Equation

(7) can also be written as:

$$\begin{aligned}
\sigma_t^{i \rightarrow j}(a^j) &= \frac{\kappa_t^{i \rightarrow j}(a^j)}{t + \sum_{a^j \in A^j} \kappa_1^{i \rightarrow j}(a^j)} \\
&= \frac{\kappa_{t-1}^{i \rightarrow j}(a^j) + \mathbb{I}_{a_{t-1}^j = a^j}}{t + 1} \\
&= \frac{t \cdot \kappa_{t-1}^{i \rightarrow j}(a^j)}{t \cdot (t + 1)} + \frac{\mathbb{I}_{a_{t-1}^j = a^j}}{t + 1} \\
&= \frac{t}{t + 1} \sigma_{t-1}^{i \rightarrow j}(a^j) + \frac{\mathbb{I}_{a_{t-1}^j = a^j}}{t + 1} \\
&= (1 - \frac{1}{t + 1}) \sigma_{t-1}^{i \rightarrow j}(a^j) + \frac{1}{t + 1} \mathbb{I}_{a_{t-1}^j = a^j}.
\end{aligned} \tag{8}$$

In addition to FP, our probabilistic model checker also supports the following learning algorithms: Geometric fictitious play (GFP) [5], and Adaptive forgetting factor fictitious play (AFFFP) [19]. These algorithms can be classified into the same category: *Algorithms based on weighted average history of actions*. They take into account all the history of players' actions in order to estimate their opponents' strategies. Actions are then selected based on these estimations. The differences of these algorithms are based on the impact of the recently observed action on the estimation of the other players' strategies. They increase the impact of the recent observation using discount factors when estimating σ^{-i} .

Example. Given the initial estimation in Equation (9) for the game in (4),

$$\kappa_0^{1 \rightarrow 2} = [0.511, 0.489]^T \text{ and } \kappa_0^{2 \rightarrow 1} = [0.489, 0.511], \tag{9}$$

we use smooth best response in Equation (2) with $\tau = 10^{-2}$ to compute the initial joint strategies, and use best response thereafter. When the game is started, we obtain that $\sigma_0^{1 \rightarrow 2} = \sigma_0^{-1} = \kappa_0^{1 \rightarrow 2}$, $\sigma_0^{2 \rightarrow 1} = \sigma_0^{-2} = \kappa_0^{2 \rightarrow 1}$. Using the above weights, equation (2) and $\tau = 0.01$ we obtain $\bar{\sigma}_0^1 = [0.9, 0.1]^T$ and $\bar{\sigma}_0^2 = [0.1, 0.9]$. This means that player 1 has probability to choose actions b_1 and b_2 with probability 0.9 and 0.1 respectively, and player 2 chooses action a_1 and a_2 with probability 0.1 and 0.9 respectively.

The initial joint strategy is shown in the ellipse node in Figure 1. It means that the probability of executing (b_1, a_1) , (b_1, a_2) , (b_2, a_1) and (b_2, a_2) is 0.09, 0.81, 0.01 and 0.09 respectively. If the players execute (b_1, a_1) , then they update their weight function using Equation (6) and obtain

$$\kappa_1^{1 \rightarrow 2} = [1.511, 0.489]^T \text{ and } \kappa_1^{2 \rightarrow 1} = [1.489, 0.511]. \tag{10}$$

The new joint strategy generated using Equations (7) and (1) is shown in the top left rectangular node in Figure 1.

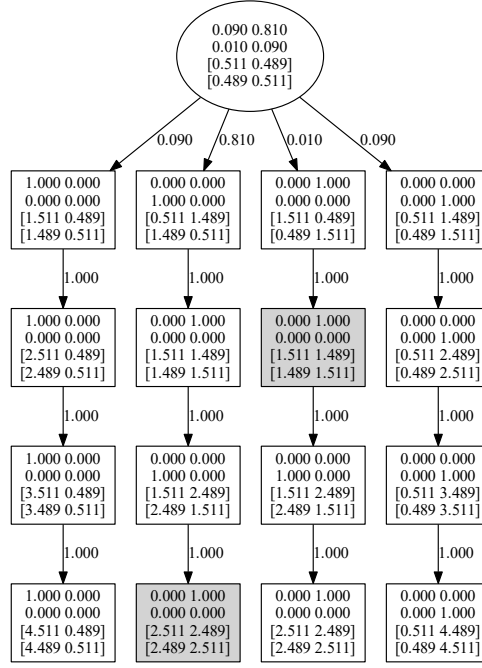


Figure 1: Five layer expansions of possible simulation traces.

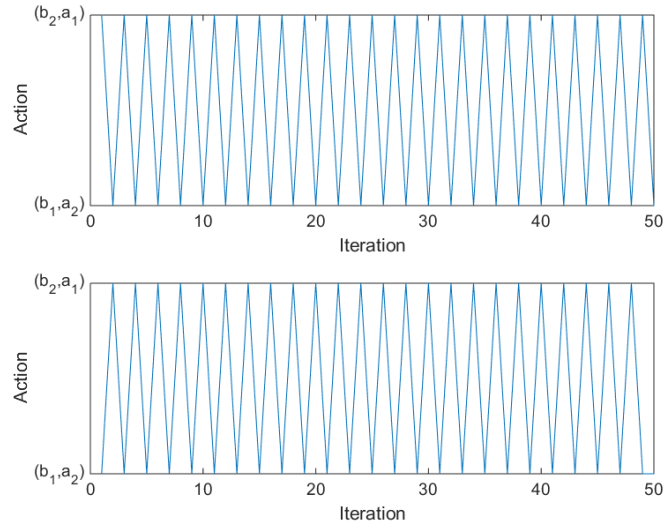


Figure 2: Simulations for the game in Equation (4)

Figure 2 are examples showing that it is difficult or impossible to understand agents' behaviour from simulation. Figure 2 shows that FP being trapped in a cycle when the game is repeatedly played for the initial conditions in Figure 1. The top figure illustrates the selected actions when the branch with initial probability 0.81 is expanded and the bottom figure the selected actions when the branch with initial probability 0.01 is expanded. In this simple case, we can observe that the two simulation runs are similar. The same cycle of actions is repeated in both figures, but they have one iteration lag. Thus, in the top figure joint action (b_1, a_2) is played in the odd iterations, while in the bottom figure it is played in the even iterations. In more complicated cases, where joint actions are repeated with lag greater than one, or the lag is unknown, it would be hard or even infeasible to determine if two simulation runs are similar.

2.3 Other learning algorithms

Geometric Fictitious Play. In FP all the previous actions are of the same importance because of the assumption that opponents' strategies are stationary. Geometric fictitious play (GFP) is a variant of FP that addresses this incorrect assumption by exponentially decaying the importance of the historical data, by a factor α ($0 < \alpha < 1$). Thus, the recently observed actions are more important in the estimation of the opponents' strategies, which is based on the following formula:

$$\sigma_t^{i \rightarrow j}(a^j) = (1 - \alpha)\sigma_t^{i \rightarrow j}(a^j) + \alpha \mathbb{I}_{a_{t-1}^j = a^j}, \quad (11)$$

where $\mathbb{I}_{a_{t-1}^j = a^j} = \begin{cases} 1 & \text{if } a_{t-1}^j = a^j \\ 0 & \text{otherwise.} \end{cases}$

Adaptive Forgetting Factor Fictitious Play. Even though GFP addresses the incorrect assumption of FP that the strategies of other players are constant, it requires the players to know in advance the rate the other players change their strategy since it uses a constant α throughout the game. In order to overcome this limitation, adaptive forgetting factor fictitious play (AFFFP) was introduced in [19]. In this variant of FP there is a time varying discount factor λ_{t-1} which is adjusted based on the likelihood of the previously observed actions. Opponents' strategies are estimated as in Equation (7), but the weights of opponents strategies are discounted as follows:

$$\kappa_t^{i \rightarrow j}(a^j) = \lambda_{t-1}^j \kappa_{t-1}^{i \rightarrow j}(a^j) + \mathbb{I}_{a_{t-1}^j = a^j} \quad (12)$$

where $\mathbb{I}_{a_{t-1}^j = a^j}$ is the same identity function as in Equation (6). Let $n_t^j = \sum_{a^j \in A^j} \kappa_t^{i \rightarrow j}(a^j)$ be the normalisation divisor in Equation (7). Based on (12), we can use the following recursion to evaluate n_t^j :

$$n_t^j = \lambda_{t-1}^j n_{t-1}^j + 1,$$

where

$$\lambda_t = \lambda_{t-1} + \gamma \left(\frac{1}{\kappa_{t-1}^{i \rightarrow j}(a)} \frac{\partial}{\partial \lambda} \kappa_{t-1}^{i \rightarrow j}(a) - \frac{1}{n_{t-1}^j} \frac{\partial}{\partial \lambda} n_{t-1}^j \right),$$

where $\gamma \in (0, 1]$ is a learning rate parameter which controls the impact of the observed actions in the new value of the adaptive factor λ ,

$$\frac{\partial}{\partial \lambda} \kappa_t^{i \rightarrow j}(a)|_{\lambda=\lambda_{t-1}} = \kappa_{t-1}^{i \rightarrow j}(a) + \lambda_{t-1} \frac{\partial}{\partial \lambda} \kappa_{t-1}^{i \rightarrow j}(a)|_{\lambda=\lambda_{t-1}}$$

and

$$\frac{\partial}{\partial \lambda} n_t^j|_{\lambda=\lambda_{t-1}} = n_{t-1}^j + \lambda_{t-1} \frac{\partial}{\partial \lambda} n_{t-1}^j|_{\lambda=\lambda_{t-1}}.$$

2.4 Discrete-Time Markov Chains

Given a set S , let $\text{Dist}(S)$ be the set of all discrete probability distributions over S , each of which is of the form $p : S \rightarrow [0, 1]$ such that $\sum_{s \in S} p(s) = 1$.

Definition 2 A DTMC M is a tuple $\langle S, \dot{s}, T, \mathcal{P}, L \rangle$, where

- S is a set of states,
- $\dot{s} \in S$ is the initial state,
- $T : S \rightarrow \text{Dist}(S)$ is a probabilistic transition relation among states. We often write $T_s(s')$ as the probability of moving from state s to state s' ,
- \mathcal{P} is a set of atomic propositions, or simply proposition.
- $L : \mathcal{P} \rightarrow 2^S$ is a labelling function that maps a proposition to a subset of states. A proposition $\mathbf{p} \in \mathcal{P}$ holds in state $s \in S$ iff $s \in L(\mathbf{p})$.

A finite path ρ is a sequence of states $s_0 s_1, \dots, s_n$ such that $T_{s_i}(s_{i+1}) > 0$ for any $0 \leq i < n$. The path ρ is infinite when $n \rightarrow \infty$. Various properties in DTMCs can be verified by probabilistic model checking. In this paper, we are interested in *steady state* properties, which specify the behaviour of a DTMC in the long run, and are captured by *bottom strongly connected components* (BSCC).

Definition 3 A set of states $\bar{S} \subseteq S$ in a DTMC M is a strongly connected component (SCC) iff the following condition holds:

$$\begin{aligned} \forall s, s' \in \bar{S} : \text{ there exists a path } \rho = s_0 s_1, \dots, s_n \\ \text{ such that } s = s_0 \text{ and } s' = s_n. \end{aligned} \quad (13)$$

Definition 4 A set of states $\bar{S} \subseteq S$ in a DTMC M is a BSCC iff it is an SCC with the following constraint:

$$\forall s \in \bar{S} \text{ and } s' \in S : T_s(s') > 0 \text{ implies that } s' \in \bar{S}. \quad (14)$$

Intuitively, a BSCC, *b SCC* hereafter, is a special strongly connected component of M such that no states outside *b SCC* can be reached from any states inside *b SCC* via the probabilistic transition relation. It means that once the system enters *b SCC*, it is trapped in it and cannot escape. BSCCs in a DTMC can be identified by applying the Tarjan’s algorithm [22] to partition the DTMC into SCCs first and then looking for SCCs that do not have outgoing transitions. The model checking algorithms for computing steady-state probabilities from BSCCs can be found in [16].

Theorem 1 *A Nash equilibrium is a BSCC.*

Proof 1 *Consider a state s that represents the joint strategy σ which is Nash equilibrium. If we assume that s is not in a BSCC, then a new state \tilde{s} will be reached with a non-zero probability that will represent a new joint strategy $\tilde{\sigma}$. This is a contradiction because a joint strategy σ is a Nash equilibrium iff no player will deviate from σ .*

Note here that a BSCC is not necessarily a Nash equilibrium. A well known example [5] comes from the FP algorithm, when it is used in the game in Equation (4). It has been shown that FP can fail to converge in the Nash equilibrium in this game and be trapped in a cycle between the actions (b_2, a_1) and (b_1, a_2) , which is also depicted in Figure 3(a). This is a BSCC, but not a Nash equilibrium. For a Pareto efficient equilibrium, all states in the BSCC have the same reward. However, it is not true for a non-Pareto efficient one. In this case, we are interested in the probability by which the system stays in some of the states in the BSCC. In the example defined in Equation (4), the Pareto efficient equilibria are captured by single-state BSCCs that fire (b_1, a_1) or (b_2, a_2) only, i.e., with probability 1.

3 Analysis of game-theoretic learning algorithms

We aim to build a framework for performance evaluation of game-theoretic learning algorithms. It explores all states an algorithm can visit. Each state contains a joint strategy σ generated from the Cartesian product of the individual strategy of all players, and extra information π , which varies among the learning algorithms.

3.1 State space generation

Definition 5 *A state s is a tuple $\langle \sigma, \pi \rangle$, where*

- σ is the joint strategy of all players, which indicates the likelihood of firing a joint action,
- π is a vector of parameters, including weights and other input variables for the learning algorithm.

Let $\kappa^i = \{\kappa^{i \rightarrow j} \mid j \in \{1, \dots, \mathcal{I}\} \setminus \{i\}\}$ be the weights that player i holds about all its opponents. The vector π for each learning algorithm is specified as follows:

- *FP*: $\pi = \langle \kappa^1, \dots, \kappa^{\mathcal{I}} \rangle$,
- *GFP*: $\pi = \langle \kappa^1, \dots, \kappa^{\mathcal{I}}, \alpha \rangle$,
- *AFFFP*: $\pi = \langle \kappa^1, \dots, \kappa^{\mathcal{I}}, n_1, \dots, n_{\mathcal{I}}, \lambda, \frac{\partial}{\partial \lambda} \kappa^1, \dots, \frac{\partial}{\partial \lambda} \kappa^{\mathcal{I}}, \frac{\partial}{\partial \lambda} n_1, \dots, \frac{\partial}{\partial \lambda} n_{\mathcal{I}} \rangle$,

Given a game and an algorithm, our probabilistic model checker adopts Breadth-First Search (BFS) to explore the state space from the initial state $\dot{s} = \langle \dot{\sigma}, \dot{\pi} \rangle$ by firing each joint action with non-zero firing probability and computing a new state. The exploration algorithm is shown in Algorithm 1. As we are interested in short-term behaviour, an upper bound on the depth in BFS can be set to terminate the exploration if it does not stop within the bound. If this is the case, then all unexplored states have a transition to a special termination state \perp with probability 1.

Algorithm 1 State space exploration

```

1:  $s_0 := \langle \dot{\sigma}, \dot{\pi} \rangle$ ;  $S := \{s_0\}$ ; enqueue(queue1,  $s_0$ )
2:  $depth := 0$ ;  $i := 1$ 
3: while queue1  $\neq \emptyset$  do
4:    $(s = \langle \sigma, \pi \rangle) :=$  dequeue(queue1)
5:   for all  $a \in A$  do
6:     if  $\sigma(a) > 0$  then
7:        $\langle \sigma'', \pi' \rangle :=$  UpdateEstimations( $\pi, a$ )
8:        $\sigma' :=$  UpdateProbDist( $u, \sigma'', \tau$ )
9:        $s_i := \langle \sigma', \pi' \rangle$ ;  $found := false$ 
10:      for all  $j = i - 1, \dots, 1$  do
11:        if  $s_j \preceq s_i$  then
12:           $T_s(s_j) := \sigma(a)$ ;  $found := true$ ; break
13:      if  $found = false$  then
14:         $T_s(s_i) := \sigma(a)$ ;  $S := S \cup \{s_i\}$ 
15:        enqueue(queue2,  $s_i$ );  $i := i + 1$ 
16:   queue1 := queue2; queue2 :=  $\emptyset$ ;  $depth := depth + 1$ 
17:   if  $depth \geq \text{UpperBound}$  then
18:      $S := S \cup \{\perp\}$ ;  $T_{\perp}(\perp) := 1$ 
19:     for all  $s \in queue1$  do
20:        $T_s(\perp) := 1$ 
21:     break
22: return ( $S, T$ )
```

In Algorithm 1, *queue1* is used to store states that need to be expanded, and *queue2* stores their success states. This is an easy way to

count the depth of the exploration. $\text{Enqueue}(\text{queue}, s)$ appends s to the tail of queue and $s := \text{dequeue}(\text{queue})$ removes the head of queue and save it to s . $\text{UpdateEstimations}(\pi, a)$ performs the estimation steps of the learning algorithms, and $\text{UpdateProbDist}(u, \sigma'', \tau)$ is the smooth best response decision rule defined in Equation (2) or best response in Equation (1). The probability of a transition executing action a comes from $\sigma(a)$.

In principle, there could be a large, even infinite, number of states for certain algorithms due to the way of updating the parameters of these algorithms. For example, the simulation in Figure 1 can be extended to infinite iterations. However, the same behaviour can be observed from the two grey nodes in this figure according to Equation (6) and (7), although these two nodes are not the same because of the weight functions. Thus, these nodes share some similarity between them, and there is no need to explore both nodes because of the same behaviour. Exploration of one node is sufficient. This observation leads to the key novelty in our stochastic model checker: *behaviour similarity relation* among states, which always results in a succinct model and makes the verification possible.

3.2 Behaviour similarity relation

Definition 6 For two state $s_1 = \langle \sigma_1, \pi_1 \rangle$ and $s_2 = \langle \sigma_2, \pi_2 \rangle$ such as s_1 is generated earlier than s_2 by Algorithm 1 such that s_1 is generated at the t_1 -th iteration and s_2 at the t_2 -th iteration ($t_1 \leq t_2$), s_1 is behaviour similar to s_2 under FP, GFP and AFFFP with best response, denoted as $s_1 \preceq s_2$, if the following condition holds:

$$s_1 \preceq s_2 \equiv (\sigma_1 = \sigma_2) \wedge \left(\bigwedge_{1 \leq i \in \mathcal{I}} r_{1,t_1}^i \preceq r_{2,t_2}^i \right), \quad (15)$$

where r_{k,t_k}^i ($k = 1, 2$) is the expected rewards that player i has over its actions A^i .

Let σ_k^i ($k = 1, 2$) be the strategy of player i in state s_k , $a_{t_k}^i \in A^i$ the action chosen to be executed in s_k , and $a_{t_k} \in A$ the joint action executed in s_k . Note that $a_{t_1} = a_{t_2}$, and therefore, $a_{t_1}^i = a_{t_2}^i$. Given a path $\rho = s'_0 \cdots s'_m$ such that s'_0 is obtained at the t'_0 -th iteration, let $\omega(\rho) = a_{t'_0} \cdots a_{t'_0+m-1}$ be the sequence of joint actions such that $a_{t'_0}$ is executed in s'_j ($0 \leq j \leq m-1$). Let \check{s}_k be the predecessor of s_k , $\check{\sigma}_k$ the probability distribution in \check{s}_k , and \check{a}_k the joint action selected in \check{s}_k . We define $r_{1,t_1}^i \preceq r_{2,t_2}^i$ iff all the following conditions holds.

(6.1)

$$\begin{aligned} \operatorname{argmax}_{a^i \in A^i} \quad & r_{1,t_1}^i(a^i) - r_{1,t_1-1}^i(a^i) = \\ \operatorname{argmax}_{a^i \in A^i} \quad & r_{2,t_2}^i(a^i) - r_{2,t_2-1}^i(a^i) \end{aligned} \quad (16)$$

(6.2) If $\check{\sigma}_1 = \check{\sigma}_2 = \sigma_1$, then $\forall a^i \in A^i$ such that $\sigma_k^i(a^i) = 0$, we either have

$$r_{2,t_2}^i(a^i) - r_{1,t_1}^i(a^i) \leq r_{2,t_2}^i(a_{t_2}^i) - r_{1,t_1}^i(a_{t_1}^i) \quad (17)$$

or

$$r^i(a_{t_1}^i, a_{t_1}^{-i}) = \max_{a^i \in A^i} r^i(a^i, a^{-i}). \quad (18)$$

(6.3) If there is a path $\rho = s'_0 \cdots s'_n$ such that $s'_0 = s_1$ and $s'_n = s_2$ and $n \geq 1$, then $\forall j \in \{1, \dots, n\}$,

$$\sigma_{1,j} = \sigma_{2,j} \wedge \bigwedge_{1 \leq i \leq \mathcal{I}} r_{2,t_2}^i(a_{t_2+j}^i) \leq r_{1,t_1}^i(a_{t_1+j}^i), \quad (19)$$

when FP is used

$$\sigma_{1,j} = \sigma_{2,j} \wedge \bigwedge_{1 \leq i \leq \mathcal{I}} r_{2,t_2}^i(a_{t_2+j}^i) \geq r_{1,t_1}^i(a_{t_1+j}^i), \quad (20)$$

when GFP and AFFFP is used

where $\sigma_{k,j}$ ($k = 1, 2$) is the joint strategy obtained by executing a sequence of joint actions $\omega(s'_0 \cdots s'_j)$ from s_k .

(6.4) If $n = 1$, then

$$\bigwedge_{1 \leq i \leq \mathcal{I}} r_{2,t_2}^i(a_{t_2}^i) \geq r_{1,t_1}^i(a_{t_1}^i). \quad (21)$$

(6.5) If there is no path from s_1 to s_2 , then $\check{\sigma}_1 = \check{\sigma}_2$ and

$$\bigwedge_{1 \leq i \in \mathcal{I}} r_{2,t_2}^i(a_{t_2}^i) \geq r_{1,t_1}^i(a_{t_1}^i), \quad (22)$$

when FP is used

$$\bigwedge_{1 \leq i \in \mathcal{I}} \left(r_{2,t_2}^i(a_{t_2}^i) \geq r_{1,t_1}^i(a_{t_1}^i) \wedge \bigwedge_{a^i \in A^i, a^i \neq a_{t_1}^i} r_{1,t_1}^i(a^i) < r_{2,t_2}^i(a^i) \right), \quad (23)$$

when GFP and AFFFP is used.

3.3 Linearity of expected reward in FP type algorithms.

In this subsection, we show the linear relation of the expected rewards of a player when Fictitious Play based learning algorithms are used. The results of this section will be used in developing the behaviour similarity relation for these algorithms.

Given two states s_1 and s_2 with $\sigma_1 = \sigma_2$, where the same action a is played in both states of s_1 and s_2 . Assume s_1 and s_2 are generated at the t_1 -th and t_2 -th iteration respectively with $t_1 \leq t_2$. Updates of player i 's estimations of its opponents' strategies in FP, GFP, and AFFFP will be the following, $\forall j \in \mathcal{I}$:

- Fictitious play updates:

$$\begin{aligned}\sigma_{t_1+1}^{i \rightarrow j} &= \begin{cases} (1 - \frac{1}{t_1+2})\sigma_{t_1}^{i \rightarrow j}(a^j) + \frac{1}{t_1+2} & \text{if } a^j \in a \\ (1 - \frac{1}{t_1+2})\sigma_{t_1}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases}, \\ \sigma_{t_2+1}^{i \rightarrow j} &= \begin{cases} (1 - \frac{1}{t_2+2})\sigma_{t_2}^{i \rightarrow j}(a^j) + \frac{1}{t_2+2} & \text{if } a^j \in a \\ (1 - \frac{1}{t_2+2})\sigma_{t_2}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases} \end{aligned} \quad (24)$$

- Geometric fictitious play updates:

$$\begin{aligned}\sigma_{t_1+1}^{i \rightarrow j} &= \begin{cases} (1 - \alpha)\sigma_{t_1}^{i \rightarrow j}(a^j) + \alpha & \text{if } a^j \in a \\ (1 - \alpha)\sigma_{t_1}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases}, \\ \sigma_{t_2+1}^{i \rightarrow j} &= \begin{cases} (1 - \alpha)\sigma_{t_2}^{i \rightarrow j}(a^j) + \alpha & \text{if } a^j \in a \\ (1 - \alpha)\sigma_{t_2}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases} \end{aligned} \quad (25)$$

- Adaptive forgetting factor fictitious play updates:

$$\begin{aligned}\sigma_{t_1+1}^{i \rightarrow j} &= \begin{cases} (1 - \alpha_{t_1})\sigma_{t_1}^{i \rightarrow j}(a^j) + \alpha_{t_1} & \text{if } a^j \in a \\ (1 - \alpha_{t_1})\sigma_{t_1}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases}, \\ \sigma_{t_2+1}^{i \rightarrow j} &= \begin{cases} (1 - \alpha_{t_2})\sigma_{t_2}^{i \rightarrow j}(a^j) + \alpha_{t_2} & \text{if } a^j \in a \\ (1 - \alpha_{t_2})\sigma_{t_2}^{i \rightarrow j}(a^j) & \text{if } a^j \notin a \end{cases} \end{aligned} \quad (26)$$

The expected reward of each action a^i for state s_i ($i = 1, 2$) is computed as follows when joint action a has been played:

$$r_{t_i}^i(a^i, \sigma_{t_i}^{-i}(a^{-i})) = \sum_{a^j \in A^{-i}} (r(a^i, a^j) \times \prod_{k: a^k \in a^j} \sigma_{t_i}^{i \rightarrow k}(a^k)). \quad (27)$$

The elements of $\prod_{k: a^k \in a^j} \sigma_{t_i}^{i \rightarrow k}(a^k)$ for FP, GFP and AFFFP are updated the above stated estimations.

The following Lemmas can be induced:

Lemma 1 Assume state s is generated in the t -th iteration of a learning process and the joint action a is played in s . Let $a^{-i}(j)$ be the action of player j ($j \neq i$) in a^{-i} , and $\sigma_t^{i \rightarrow j}(a^{-i}(j))$ the element of $\sigma_t^{i \rightarrow j}$ which corresponds to the action $a^{-i}(j)$. For all actions in $A^i \setminus \{a^{-i}(j)\}$, the rank among them in $\sigma_t^{i \rightarrow j}$ is maintained in $\sigma_{t+1}^{i \rightarrow j}$, i.e., if $\sigma_t^{i \rightarrow j}(a_1^j) > \sigma_t^{i \rightarrow j}(a_2^j)$, then $\sigma_{t+1}^{i \rightarrow j}(a_1^j) > \sigma_{t+1}^{i \rightarrow j}(a_2^j)$, $\forall a_1^j, a_2^j \in A^j \setminus \{a^{-i}(j)\}$.

Proof 2 Player i updates its estimates about the strategies of its opponents using a^{-i} . Then $\sigma_{t+1}^{i \rightarrow j}(a^{-i}(j))$ will be increased by $\frac{1}{t}(1 - \sigma_t^{i \rightarrow j}(a^{-i}(j)))$ and for all $a^j \in A^j \setminus \{a^{-i}(j)\}$, the estimate $\sigma_{t+1}^{i \rightarrow j}(a^j)$ will be decreased by $\frac{1}{t}\sigma_t^{i \rightarrow j}(a^j)$. For any two numbers x and y , we have $x > y \wedge t > 0 \Leftrightarrow (1 - \frac{1}{t})x > (1 - \frac{1}{t})y$.

Lemma 2 Assume that the same joint action a is played in any two consecutive iterations, i.e., t -th and $(t + 1)$ -th iterations, using FP, GFP or AFFFP. Given two actions $x^i \in A^i$ and $y^i \in A^i$ ($y^i \neq x^i$) of player i of which x^i is the best response to a , let

$$\begin{aligned} rclw^\dagger &= r_{t+1}^i(x^i, \sigma_{t+1}^{-i}(x^{-i})) - r_t^i(x^i, \sigma_t^{-i}(x^{-i})), \\ w^\dagger &= r_{t+1}^i(y^i, \sigma_{t+1}^{-i}(y^{-i})) - r_t^i(y^i, \sigma_t^{-i}(x^{-i})), \\ \bar{w}^\dagger &= r_{t+2}^i(x^i, \sigma_{t+2}^{-i}(x^{-i})) - r_{t+1}^i(x^i, \sigma_{t+1}^{-i}(x^{-i})), \\ \bar{w}^\ddagger &= r_{t+2}^i(y^i, \sigma_{t+2}^{-i}(y^{-i})) - r_{t+1}^i(y^i, \sigma_{t+1}^{-i}(y^{-i})). \end{aligned}$$

We have $w^\dagger > w^\ddagger$ and $\bar{w}^\dagger > \bar{w}^\ddagger$.

Proof 3 The fact that x^i is the best response to a indicates that if $\sigma^{-i}(a^{-i}) = 1$, then player i will choose action x^i at the next iteration. If $\sigma^{-i}(a^{-i}) < 1$, then there exists a threshold $0 < f < 1$ such that player i will continue to choose action a^i when $\sigma^{-i}(a^{-i}) < f$; player i will choose x^i when $\sigma^{-i}(a^{-i}) \geq f$ [5]. As the expected rewards are linear functions, we have $\bar{w}^\dagger > 0$, which means that player i 's confidence on selecting x^i is increasing as a is played again. This confidence will keep increasing as $\sigma^{-i}(a^{-i})$ increases, until $\sigma^{-i}(a^{-i}) \geq f$ and then x^i is played. If there exists another action $y^i \neq x^i$ whose expected reward is also increased at the $(t + 1)$ -th iteration, then we have $w^\dagger > 0$. If its expected reward is increasing faster than x^i , i.e., $w^\dagger < w^\ddagger$ and $\bar{w}^\dagger < \bar{w}^\ddagger$, then at a later iteration of any of the game playing process FP, GFP and AFFFP, it is possible to select y^i if a is played sufficient number of times to reach $\sigma^{-i}(a^{-i}) \geq f$. However, this is a contradiction because the best response to a is x^i and not y^i . Furthermore, we cannot have $w^\dagger < w^\ddagger \wedge \bar{w}^\dagger \geq \bar{w}^\ddagger$ or $w^\dagger \geq w^\ddagger \wedge \bar{w}^\dagger < \bar{w}^\ddagger$ because of the expected rewards are linear functions with respect to the estimates of other players' strategies.

3.4 State space reduction

Now we can explain what *similarity* means for FP based learning algorithms. First, we need to define *BSCC actions* as follows. Let σ_s denote that the strategy σ in a state $s = \langle \sigma, \pi \rangle$.

Definition 7 Given a BSCC $bscc$, the corresponding BSCC actions is the set of actions $A_{bscc} \subseteq A$ such that

$$A_{bscc} = \{a \in A \mid \exists s \in bscc \text{ such that } \sigma_s(a) > 0\}.$$

Theorem 2 *Given two states s_1 and s_2 such that $s_1 \preceq s_2$ under FP, GFP, and AFFFP, if a set of BSCC actions A_{bscc} can be reached from s_1 , then A_{bscc} will be reached from s_2 .*

Proof 4 *In order to complete this proof it is needed to show that when two states, s_1 and s_2 are found to be similar then the same actions will be selected in their successor states and eventually they will converge to the same BSCC. We should notice here that it is not necessary for both states to need the same number of successor states in order to reach a BSCC.*

Suppose that s_1 is generated at iteration t_1 and s_2 at t_2 and we have $t_1 \leq t_2$. For all the variants of FP, that s_1 and s_2 are found similar belong to one of the following four cases:

1. *There is no path from s_1 to s_2 and s_1 itself is a BSCC;*
2. *There is no path from s_1 to s_2 and s_1 alone is not a BSCC;*
3. *s_2 is the direct successor state of s_1 ;*
4. *There are intermediate states between s_1 and s_2 .*

Case 1:

As s_1 is a single state BSCC, the joint action a selected in s_1 will be played infinitely from s_1 onward, which implies that a Nash equilibrium is reached by playing a . By definition of behaviour similarity, the same joint action a will be also played in s_2 , and therefore, the same Nash equilibrium is reached. By the definition of Nash equilibrium, we know that from s_2 , action a will be played permanently.

Case 2:

Similarly to Case 1 since $s_1 \preceq s_2$, conditions (6.1), (6.2) and (6.5) should be satisfied. This case can be divided in two sub-cases. The first one consists of the instances where $\check{\sigma}_1 = \check{\sigma}_2 \neq \sigma_1$, which means that the joint actions \check{a}_k ($k = 1, 2$) from the predecessors of s_1 and s_2 are the same, but they are different from the action a_{t_k} selected in s_1 and s_2 . In both s_1 and s_2 , the selected joint action changes from \check{a}_k to its best response a_{t_k} , which denotes that the players in both states reach the necessary confidence level in order to change actions [2, 5]. Since a_{t_k} is not a Nash equilibrium, there exists another joint action \bar{a}_{t_k} that is the best response to a_{t_k} . Furthermore, we have $\bar{a}_{t_1} = \bar{a}_{t_2}$ due to $a_{t_1} = a_{t_2}$ and condition (6.5). This condition is used to guarantee that the selected action is not affected by smooth best response at the initial iteration. As the best response $\bar{a}_{t_k}^i$ to $a_{t_k}^i$ for player i is deterministic, if the joint best response \bar{a}_{t_1} is not equal to \bar{a}_{t_2} , then we know that there exists a player j that excises its best response earlier than some other players. This asynchronous change is caused by the smooth best response at the initial iteration: players chooses a non-best response to their initial estimation. Condition (6.5) is designed to prevent two states

from being classified similar when asynchronous changes can occur in their successor states. Note here that in the case of GFP and AFFFP, the learning rate will be the same in both s_1 and s_2 , as it does not depend on the number of iterations. For that reason, the extra constraint in Equation (24) is needed. When the joint action \bar{a}_{t_k} is selected, the estimates of opponents strategies will be increased for the elements that participate in the joint action \bar{a}_{t_k} . Players will learn that \bar{a}_{t_k} is selected and eventually they will select the best response to \bar{a}_{t_k} , say $\bar{a}_{t_k}^i$, simultaneously. This process will continue until a Nash equilibrium or a BSCC is reached.

In the second case, the same action has been played in s_1 , s_2 and their predecessors. Similarly to the first case because of condition (6.1), there is another action $\bar{a}_{t_k}^i$ that is best response to $a_{t_k}^i$. Nonetheless, it not necessary that the players will simultaneously change their action and result to the same joint action from both states. Assume that exists another action b^i in state s_2 such as $r_{2,t_2}^i(a_{t_k}^i) - r_{2,t_2}^i(b^i) < r_{2,t_2}^i(a_{t_k}^i) - r_{2,t_2}^i(\bar{a}_{t_k}^i)$. This implies that it is possible for $r_{2,t_2}^i(b^i)$ to exceed $r_{2,t_2}^i(a_{t_k}^i)$ in fewer iterations than $r_{2,t_2}^i(\bar{a}_{t_k}^i)$ will do. Condition (6.2) is designed to overcome this problem. This condition ensures that the difference in expected reward between the selected and the non-selected actions in s_2 , which has the slowest learning rate due to $t_2 \geq t_1$, is smaller than the one in s_1 . Condition (6.2) implies that $r_{1,t_1}^i(a_{t_k}^i) - r_{1,t_1}^i(b^i) < r_{2,t_2}^i(a_{t_k}^i) - r_{2,t_2}^i(b^i)$. Therefore, if b^i was to be selected from a successor of s_2 , then the expected reward of b^i should be increased more from s_2 than from s_1 . Therefore, because of Lemma 1 and the Condition (6.5), it is not possible for b^i to be selected from a successor of s_2 given that it was not selected from a successor of s_1 .

Now we assume that \bar{a}_{t_k} has been played in a successor state of s_1 and s_2 at the $(t_1 + n_1)$ -th and $(t_2 + n_2)$ -th iterations respectively. We have the following inequalities:

$$r_{1,t_1+n_1}^i(\bar{a}_{t_k}^i) > r_{1,t_1+n_1}^i(a_{t_k}^i) \quad r_{2,t_2+n_2}^i(\bar{a}_{t_k}^i) > r_{2,t_2+n_2}^i(a_{t_k}^i).$$

If \bar{a}_{t_k} is not a Nash equilibrium, we assume that $\bar{a}^i \in A^i$ is the action of robot i that has the highest increment in the expected rewards such that

$$r_{1,t_1+n_1}^i(\bar{a}^i) < r_{1,t_1+n_1}^i(\bar{a}_{t_k}^i) \quad r_{2,t_2+n_2}^i(\bar{a}^i) < r_{2,t_2+n_2}^i(\bar{a}_{t_k}^i).$$

If the expected reward of action $c^i \in A^i$ is also increasing, then we have:

$$r_{1,t_1+n_1}^i(c^i) < r_{1,t_1+n_1}^i(\bar{a}_{t_k}^i) \quad r_{2,t_2+n_2}^i(c^i) < r_{2,t_2+n_2}^i(\bar{a}_{t_k}^i).$$

Assume that \bar{a}^i will be selected at a successor state of s_1 . Therefore the expected reward of action \bar{a}^i will exceed the expected reward of $\bar{a}_{t_k}^i$ either in fewer iterations than that of c^i does, or if they need the same number of iterations, then its expected reward will be greater than that of c^i . Based on the linearity of the expected reward with respect to the expected strategy,

Lemma 1, Lemma 2 and the above inequalities, the same will also happen in the successor states of s_2 with \bar{a}^i being eventually selected. If \bar{a}^i is not a Nash equilibrium, then the next action will be selected from both states for the same reason until a Nash equilibrium or a BSCC is reached.

Case 3:

In this case s_2 is the immediate successor of s_1 , i.e. $t_2 = t_1 + 1$. Similarly to the previous cases since $s_1 \preceq s_2$ conditions (6.1), (6.2) and (6.4) will be satisfied. If s_1 is a BSCC then the selected action in all its successor states will be the same, e.g., a . Equation (16) and (21) show that a will be played also in the successor state of s_2 , since players use best response to select actions. Although BR does not guarantee that there will not exist another action \bar{a} that will be selected in a later state. However, if Equation (18) is met, then player i will not like to change its action, as this will be the one which will maximise its reward. If Equation (17) is satisfied, then the difference between the expected reward of selected action and the actions $a^i : a^i \in A^i \wedge a^i \notin a$ is increasing in s_2 , as Equation (17) can be written as $\forall a^i \in A^i$ such that $\sigma_k^i(a^i) = 0$:

$$r_{1,t_1}^i(a_{t_1}^i) - r_{1,t_1}^i(a^i) \leq r_{2,t_2}^i(a_{t_2}^i) - r_{2,t_2}^i(a^i).$$

Therefore, the expected reward of joint action a will be increased more than all the other joint actions, and thus, it will be selected in all the successor states of s_1 and s_2 . Thus a is a Nash equilibrium and s_1 and s_2 can be characterised as similar states.

Case 4:

In this case, conditions (6.1), (6.2) and (6.3) hold, but FP, GFP, and AFFFP are treated differently in condition (6.3). In this loop $s'_0 \cdots s'_n s'_0$, the estimations of a player's opponents strategies, i.e., Equation (24), of FP would converge to the number of occurrences each joint action appears in the loop. That is, if a joint action a appears in the loop m times over the $N = n + 1$ actions of the loop, then for each player $i \in \{1, \dots, \mathcal{I}\}$, its estimates of the opponents strategies $\sigma^{i \rightarrow j}(a^j)$ is $\frac{m}{N}$ for all $j \in \{1, \dots, \mathcal{I}\} \setminus \{i\}$ and $a^j \in a$ [5]. Thus, the fluctuations of the expected reward will vanish through time. Hence if the expected reward for a specific joint action of the loop increases, this denotes that s_1 is not a BSCC. On the other hand, GFP and AFFFP do not converge to the mixed Nash equilibrium of a game. Instead, they will converge to a mixed strategy, which depends on α and λ respectively. The expected reward for the joint action a will be greater than $\frac{m}{N}$ [5, 19]. Therefore, it is expected that the estimates of the opponents playing the observed actions will increase, as well as the specific expected rewards, although the increment will get smaller in every iteration of the loop. Thus, when Equation (20) is satisfied, the the same loop is generated from s_1 and s_2 and their successors.

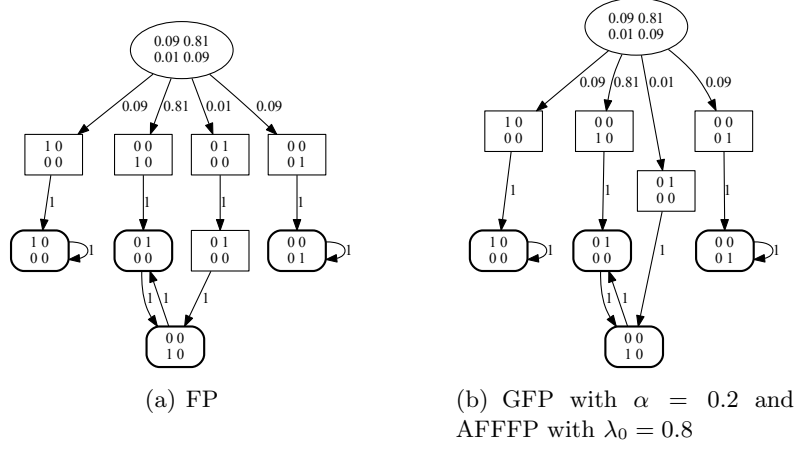


Figure 3: The DTMCs for FP, GFP and AFFFP on the simple coordination example

4 Case studies

We have implemented our method in a prototype tool. In this section we present case studies to demonstrate our performance evaluation method on several case studies.

4.1 Simple coordination game

The game described by Equation (4) is a symmetric game. In this game, players can easily be trapped in cycles composed of joint actions (b_1, a_2) and (b_2, a_1) . This cycle does not have any reward. Figures 3 illustrates the DTMCs (with weights omitted) for the FP, GFP and AFFFP algorithms for the example in Equation (4) under the initial estimation in Equation (9). In each DTMC, the ellipse nodes represent the initial states. The BSCCs are shown in nodes with rounded corners.

These figures clearly illustrate that FP, GFP and AFFFP cannot avoid this cycle.

Three measures were used to compare the performance of the algorithms: the number of states, the depth needed to reach these states, and the probability each algorithm has to converge into Pareto efficient Nash equilibria. These are three general measures that can be used in order to compare the above mentioned algorithms. Depending on the problem, more measures can be defined in our tool, such as the probability of a state of interest being reached even if it is not a Nash equilibrium, or the probability of not reaching a specific state. However, the exploration of all possible performance measures is not the focus of this work. The computational time is not reported because in the most of the cases were just a fraction of one

Table 1: Experimental results for the simple coordination game with $\tau = 1$.

Performance	FP	GFP	AFFFP
Number of states	9	64	31
Number of iterations	2	57	24
Convergence probability	0.7204	0.8313	0.6666

Table 2: Process to create the reward matrix of the complex coordination game where each player has 20 available actions to choose with $\delta = 0.001$.

· set $n = 5$, $\zeta = 1 + \frac{1}{n^{1-\delta}}$, $\beta = 1 - \frac{1}{n^{2*(1-\delta)}}$
· $u(i, j) = 1$, $\forall i \in [n+1, 4n]$, $j = i$
· $u(i, j) = 1$, $\forall i \in [2, n]$, $j = i - 1$
· $u(i, j) = \zeta$, $\forall i \in [n+1, 4n]$, $j = i - 1$
· $u(i, j) = \zeta$, $i = 2n + 1$, $j = 4n$
· $u(i, j) = \beta$, $\forall j \leq 2n$, $i > j$
· $u(i, j) = \beta$, $\forall i - j \leq n$, $i > j$
· $u(i, j) = \beta$, $\forall i \in [2n+1, j-n]$, $j \in [3n+1, 4n]$
· $u(i, j) = 0$, Otherwise

second.

In the experiments, we set $\tau = 1$ for smooth best response used in the first iteration. We generate 100 random initial estimates and run each algorithm over them to test their average performance, as the initial estimate can have strong impact on the performance. The results we obtained for these games are reported in Table 1. GFP has the highest probability to reach the pure Nash equilibria among all FP-based algorithms.

4.2 Complex coordination game

This game was introduced in [8] in order to show the asymptotic properties of FP. It is also a symmetric game where the rewards for each player is the same. The rewards are generated according to Table 2. Each player has 20 actions in this example, but it can be extended to a larger number of actions. The full reward matrix can be found in Appendix.

It was shown in [8] that for any $\delta > 0$, FP will always converge to a solution with a reward no less than ξ , where ξ is defined as the utility of the Pareto efficient Nash equilibrium subtracting a constant $\epsilon = \frac{1}{2}$. Furthermore, this result was obtained for a single initial joint action (a_1, b_1) .

In this work, we study the performance of FP under random initial conditions with $0 < \kappa_t^{i \leftarrow j}(a^j) \leq 1, \forall a^j \in A^j, \forall j \in \{1, \dots, \mathcal{I}\}$ and $\delta = 0.001$. We found that for this particular δ and initial conditions, FP almost surely converges to the Pareto efficient Nash equilibrium. This supports the importance of the proposed methodology, since even in cases where the theoretical

Table 3: Experimental results for the complex coordination game with $\tau = 1$.

Performance	FP	GFP	AFFFP
Number of states	2373	2678	2271
Number of iterations	90	7	32
Convergence probability	0.9969	0.999	0.9115

properties of a game are well known, there are specific conditions which can be of interest in practical applications where the outcome of FP will defer.

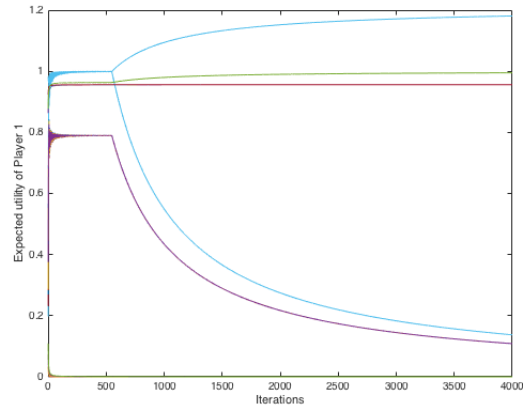
Additionally, under specific initial conditions there is a temporary cycle between two joint actions, e.g., $a^{c1} = (b_{20}, a_6)$ and $a^{c2} = (b_6, a_{19})$, which disappears after the t^{th} iteration. Figures 4(a) and 4(b) depict the expected reward of each action of player 1 and 2 respectively. In both figures the expected rewards of the actions in support of a^{c1} and a^{c2} are similar in the initial iterations. However, after $t > 500$ iterations, FP converge to a Pareto efficient Nash equilibrium, which is a single action whose expected reward is maximised. The reason for this behaviour is illustrated in Figures 5(a) and 5(b), where Players estimates of their opponent’s strategy are depicted after 100 and 2000 iterations respectively. In Figure 5(a), there are fluctuations in Player 1’s estimates that leads to the cycles between the joint actions. These fluctuations disappear in Figure 5(b) when FP converged to the Nash equilibrium.

This behaviour was captured by the stopping criterion we used for FP. When 100 iterations were used the proposed methodology does not produce a terminal state. Thus, based on the fluctuations of the estimates about players’ opponents’ strategies, and consecutively their expected rewards, there were no evidence of a persistent cycle or a Pareto efficient Nash equilibrium. When we increase the number of iterations to 3000, the Pareto efficient Nash equilibrium was identified as a terminal state.

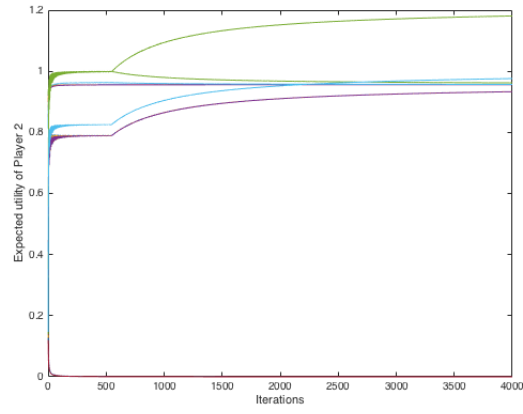
In this example, FP, GFP and AFFFP have good performance. Their convergence probability is very close to 1. From Tables 1 and 3, we conjecture that GFP has great potential to be applied in practice for its high convergence probability and fast convergence.

4.3 Shapley’s game

Games can often be classified in two categories based on the structure of their reward function: *cooperative* and *non-cooperative* games. The main difference between these two categories is that in cooperative games, there is a joint action that maximises all players’ reward, while in non-cooperative games, no joint actions can maximise all players’ reward. In the latter case,

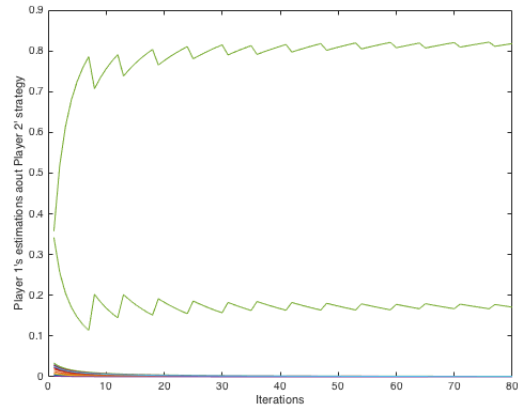


(a) Player 1

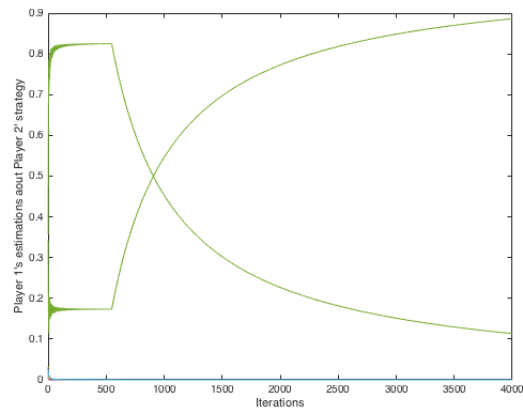


(b) Player 2

Figure 4: Expected rewards in the complex coordination game



(a) 80 iterations



(b) 400 iterations

Figure 5: Estimates of the opponent' strategy in the complex coordination game

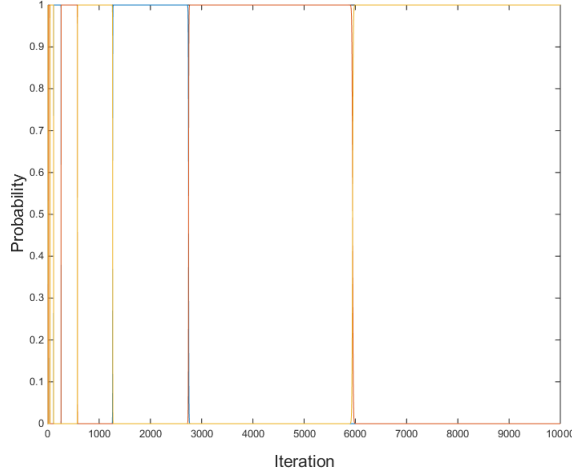


Figure 6: Simulation for Shapley's game

we selected a joint action that is “fair” for both players.

$$r = \begin{matrix} & a_1 & a_2 & a_3 \\ \begin{matrix} b_1 \\ b_2 \\ b_3 \end{matrix} & \left[\begin{array}{c|c|c} 0,0 & 1,0 & 0,1 \\ 0,1 & 0,0 & 1,0 \\ 1,0 & 0,1 & 0,0 \end{array} \right] \end{matrix} \quad (28)$$

Equation (28) shows the Shapley's game [17], which does not have any pure Nash equilibria. However, there are two mixed strategy equilibria. In the first mixed strategy equilibrium, the players choose with the same probability the joint actions (b_1, a_1) , (b_2, a_2) and (b_3, a_3) . In the second one, the players choose actions based on the following cycle:

$$\begin{aligned} (b_1, a_2) &\rightarrow (b_1, a_3) \rightarrow (b_2, a_3) \rightarrow (b_2, a_1) \\ &\rightarrow (b_3, a_1) \rightarrow (b_3, a_2) \rightarrow (b_1, a_2). \end{aligned} \quad (29)$$

Figure 6 depicts the probability of players to reach a decision in Shapley's game in Equation (28) when FP is used. In particular, each colour represent one of the joint actions that comprises the cycle in Equation (30). From Figure 6, it is reasonable to assume that FP converges to a single action because this joint action is played for more than a thousand iterations of the game. However, it is well known [17] that FP is trapped in this cycle, where each of the six joint actions in this cycle is repeated for a large, but still finite, number of iterations as $t \rightarrow \infty$. In particular, the number of repetitions for each joint action in a loop is increased as $t \rightarrow \infty$. Our tool does not terminate on this cycle because of its irregular shape.

However, our tool successfully captures the cycle in the first mixed strategy equilibrium. This cycle can only be generated when both players use

equal weights as the initial estimates, which is shown in Equation (30).

$$\kappa_0^{1 \rightarrow 2} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T \text{ and } \kappa_0^{2 \rightarrow 1} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}], \quad (30)$$

It is also worth noting that if two or more actions have the same expected rewards when computing Equation (1) in best response, which is the case in this mixed strategy equilibrium, then the first action will be chosen to execute. The DTMC showing the cycle is presented in Figure 7, where the states and transitions for the second equilibrium is omitted. Note that the prefix before the cycle in the middle and right branches in this figure is generated due to the smooth best response rule in the first iteration. Under best response and equal initial weights in Equation (30), the players would definitely choose (b_1, a_1) , i.e., the left branch in the figure, while smooth best response can force them to play (b_2, a_2) (the middle branch) and (b_3, a_3) (the right branch) with non-zero probability. These two branches enter the cycle after the weights of all actions become equal again and this time they use best response.

4.4 Discussion

The 20×20 symmetric game a game, which was introduced in [8] in order to show the poor performance of *FP*. Nonetheless their reported results are not always correct since for the specific parameters *FP* converged almost always in the NE of the game.

The experimental results are indicative of that no algorithm performed perfectly in all games initialised. The two variants of Regret Matching have better performance than the other algorithms on the simple coordination game, but much worse performance on the complex coordination game than the *FP* based algorithms. Although *GFP* has the best average performance in the two coordination games, there exist initial conditions in the complex coordination game where *GFP* fails to converge to Nash equilibrium. This suggests that there could be no single learning algorithm that is optimal for every competitive game or cooperative-game-based distributed optimisation. Therefore, a selection method for learning algorithms, such as our tool, is useful to have in practice.

5 Conclusions

This paper presented a novel approach that can evaluate the performance of game-theoretic learning algorithms over finite time. The key complexity reduction technique is the use of a *behaviour similarity relation*, which makes verification feasible by reducing the number of states. The effectiveness has been tested on a set of learning algorithms using various examples. These examples illustrated the principles of our new method as they have

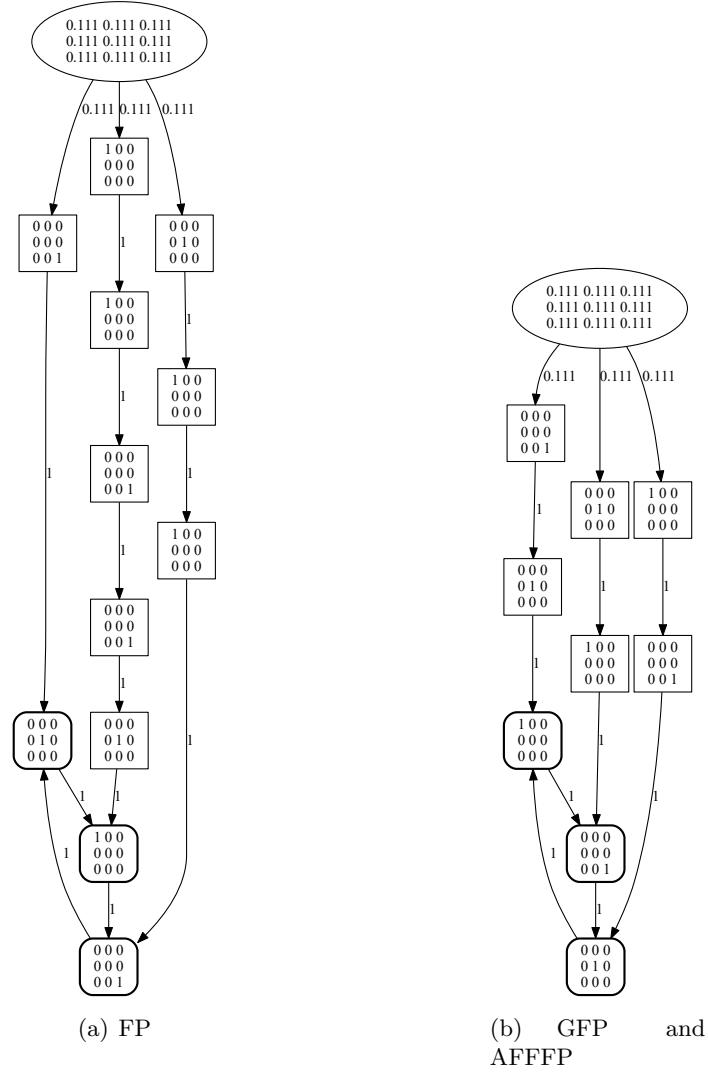


Figure 7: DTMC for the mixed strategy equilibrium in Shapley's game

relatively well-known dynamics and equilibria, which allowed us to carefully examine the experimental results and confirm their correctness. The cumulative performance result clearly demonstrated the importance of using our approach in this domain.

Future work will include investigating the application of the techniques to multi-player games and cooperative games in robotics and other real-world scenarios for distributed optimisation. Extending our approach to more learning algorithms is another direction we want to pursue. The proposed behaviour similarity relation can be used for any myopic algorithms and FP based algorithms which do not use randomisation for computing σ^{-i} . However, some FP based algorithms adopt randomisation to predict other players' strategies, such as Extended Kalman Filter Fictitious Play (EKFFP) [18] and Particle Filter Fictitious Play (PFFP) [20]. It is a challenge to define behaviour similarity relation and calculate the upper bound of maximum error for these algorithms. Another direction is to evaluate the impact of initial strategies.

Acknowledgment

This work was supported by the EPSRC project EP/J011894/2.

References

- [1] T. Ayken and J.-i. Imura, "Asynchronous distributed optimization of smart grid," in *Proc. SICE Annual Conference (SICE'12)*. IEEE, 2012, pp. 2098–2102.
- [2] G. W. Brown, "Iterative solutions of games by fictitious play," in *Activity Analysis of Production and Allocation*, T. C. Koopmans, Ed. Wiley, 1951, pp. 374–376.
- [3] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Approximate solutions for partially observable stochastic games with common payoffs," in *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*. IEEE, 2004, pp. 136–143.
- [4] —, "Game theoretic control for robot teams," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 1163–1169.
- [5] D. Fudenberg and D. Levine, *The theory of Learning in Games*, K. Binmore, Ed. The MIT Press, 1998.
- [6] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

- [7] E. Gelenbe and S. Timotheou, “Random neural networks with synchronized interactions,” *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008.
- [8] P. W. Goldberg, R. Savani, T. B. Sørensen, and C. Ventre, “On the approximation performance of fictitious play in finite games,” *Int. J. Game Theory*, vol. 42, no. 4, pp. 1059–1083, 2013.
- [9] J. Kho, A. Rogers, and N. R. Jennings, “Decentralized control of adaptive sampling in wireless sensor networks,” *ACM Transactions on Sensor Networks*, vol. 5, no. 3, p. 19, 2009.
- [10] —, “Decentralized control of adaptive sampling in wireless sensor networks,” *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 1–35, 2009.
- [11] D. S. Leslie and E. Collins, “Generalised weakened fictitious play,” *Games and Economic Behavior*, vol. 56, no. 2, pp. 285 – 298, 2006.
- [12] B. May, N. Korda, A. Lee, and D. S. Leslie, “Optimistic bayesian sampling in contextual-bandit problems,” *J. Mach. Learn. Res.*, vol. 13, pp. 2069–2106, 2012.
- [13] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, “Taming decentralized pomdps: Towards efficient policy computation for multi-agent settings,” in *IJCAI*, 2003, pp. 705–711.
- [14] J. Nash, “Equilibrium points in n-person games,” in *Proc. the National Academy of Science, USA*, vol. 36, 1950, pp. 48–49.
- [15] F. A. Oliehoek, M. T. Spaan, J. S. Dibangoye, and C. Amato, “Heuristic search for identical payoff bayesian games,” in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1115–1122.
- [16] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*. Amer. Math. Soc., 2004.
- [17] L. Shapley, *In advances in Game theory*. Princeton University, 1964.
- [18] M. Smyrnakis and S. Veres, “Coordination of control in robot teams using game-theoretic learning,” in *Proc. IFAC’14*, 2014, pp. 1194–1202.
- [19] M. Smyrnakis, “Game-theoretical approaches to decentralised optimisation,” Ph.D. dissertation, University of Bristol, 2011.
- [20] M. Smyrnakis and D. S. Leslie, “Dynamic Opponent Modelling in Fictitious Play,” *The Computer Journal*, vol. 53, pp. 1344–1359, 2010.

- [21] A. Stranjak, P. S. Dutta, M. Ebdem, A. Rogers, and P. Vytelingum, "A multi-agent simulation system for prediction and scheduling of aero engine overhaul," in *Proc. AAMAS'08*, 2008, pp. 81–88.
- [22] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, pp. 146–160, 1972.
- [23] T. Voice, P. Vytelingum, S. D. Ramchurn, A. Rogers, and N. R. Jennings, "Decentralised control of micro-storage in the smart grid." in *AAAI*, 2011, pp. 1421–1426.

Appendix: Reward matrix for the complex coordination game

	a_1	a_2	$a_3 a_4$	a_5	a_6	a_7	a_8	a_8	a_9	a_{10}
b_1	0	0	0	0	0	0	0	0	0	0
b_2	1	0	0	0	0	0	0	0	0	0
b_3	0.95594	1	0	0	0	0	0	0	0	0
b_4	0.95594	0.95594	1	0	0	0	0	0	0	0
b_5	0.95594	0.95594	0.95594	1	0	0	0	0	0	0
b_6	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0	0	0
b_7	0.95594	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0	0
b_8	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0
b_9	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	1.2099	1	0
b_{10}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	1.2099	1
b_{11}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	1.2099
b_{12}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{13}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{14}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{15}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{16}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{17}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{18}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{19}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
b_{20}	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594	0.95594
$r =$	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
b_1	0	0	0	0	0	0	0	0	0	0
b_2	0	0	0	0	0	0	0	0	0	0
b_3	0	0	0	0	0	0	0	0	0	0
b_4	0	0	0	0	0	0	0	0	0	0
b_5	0	0	0	0	0	0	0	0	0	0
b_6	0	0	0	0	0	0	0	0	0	0
b_7	0	0	0	0	0	0	0	0	0	0
b_8	0	0	0	0	0	0	0	0	0	0
b_9	0	0	0	0	0	0	0	0	0	0
b_{10}	0	0	0	0	0	0	0	0	0	0
b_{11}	1	0	0	0	0	0.95594	0.95594	0.95594	0.95594	1.2099
b_{12}	1.2099	1	0	0	0	0	0.95594	0.95594	0.95594	0.95594
b_{13}	0.95594	1.2099	1	0	0	0	0	0.95594	0.95594	0.95594
b_{14}	0.95594	0.95594	1.2099	1	0	0	0	0	0.95594	0.95594
b_{15}	0.95594	0.95594	0.95594	1.2099	1	0	0	0	0	0.95594
b_{16}	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0	0	0
b_{17}	0	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0	0
b_{18}	0	0	0.95594	0.95594	0.95594	0.95594	1.2099	1	0	0
b_{19}	0	0	0	0.95594	0.95594	0.95594	0.95594	1.2099	1	0
b_{20}	0	0	0	0	0.95594	0.95594	0.95594	0.95594	1.2099	1